Cambridge Assessment International Education

Curriculum Framework

Cambridge Primary Computing 0059

Published in September 2021 for first teaching in September 2022.

Version 1.0

Cambridge Pathway

Why choose Cambridge International?

Cambridge International prepares school students for life, helping them develop an informed curiosity and a lasting passion for learning. We are part of the University of Cambridge.

Our Cambridge Pathway gives students a clear path for educational success from age 5 to 19. Schools can shape the curriculum around how they want students to learn – with a wide range of subjects and flexible ways to offer them. It helps students discover new abilities and a wider world, and gives them skills they need for life so they can thrive throughout their schooling work and life.

Our programmes and qualifications set the global standard for international education. They are created by subject experts, rooted in academic rigour and reflect the latest educational research. They provide a strong platform for students to progress from one stage to the next, and are well supported by teaching and learning resources.

Our mission is to provide educational benefit through provision of international programmes and qualifications for school education and to be the world leader in this field. Together with schools, we develop Cambridge students who are confident, responsible, reflective, innovative and engaged – equipped for success in the modern world.

Every year, nearly a million Cambridge students from 10 000 schools in 160 countries prepare for their future with the Cambridge Pathway.

Quality management
Cambridge International is committed to providing exceptional quality. In line with this commitment, our quality management system for the provision of international qualifications and education programmes for students aged 5 to 19 is independently certified as meeting the internationally recognised standard, ISO 9001:2015. Learn more at **www.cambridgeinternational.org/ISO9001**

Contents

1 Why choose this curriculum framework?

Key benefits

Cambridge Primary Computing supports young learners to discover the technological world and to understand how many of the things that they see and use every day actually work. Learners will understand that computers and machines are not operated by magic or by another unseen human being. They will also begin to understand how the data that we input is combined with logical sequences of instruction to generate the outputs that we require from different devices.

Primary aged learners are already likely to be developing an understanding of how they are informed and entertained by a range of digital devices, through their personal explorations and through other areas of their education. The Cambridge Primary Computing curriculum takes this further by enabling them to see what happens on the inside of a computer. Learners will understand how a range of core principles, some of which are centuries old, are applied to the development of increasingly capable computers and machines, and to the services that are controlled by these devices.

Ever since computers were first introduced into classrooms and workplaces, improvements in technology have been continual, dynamic and, in many industries, have revolutionised the way that people work. Therefore, it is important that learners understand how hardware, software and computational thought processes are combined to make computers such essential and exciting parts of our lives. This curriculum therefore supports learners to:

- understand the role of each physical part of a computer system and how software drives what happens inside each of those parts
- develop logical thinking skills, including decomposition, abstraction, pattern recognition and precision.

Learners can apply these thinking skills across all areas of their education, including in Computing where they are used in the creation of computer programs, starting with simple sequences of instructions using a fun and visual programming language. Creating their own programs will support learners to increasingly understand:

- the relationship between inputs and outputs
- how to identify and solve problems, and
- how to program a computer to make decisions based upon the information that it has been given.

Learners will also represent their algorithms verbally and visually, recognising the need to be precise and concise.

Learners will also understand the role of data within computers and how computers are used to gather, store, sort and represent data within spreadsheets and other databases. They will consider how data is transferred between devices and the risks that are associated with data transfer.

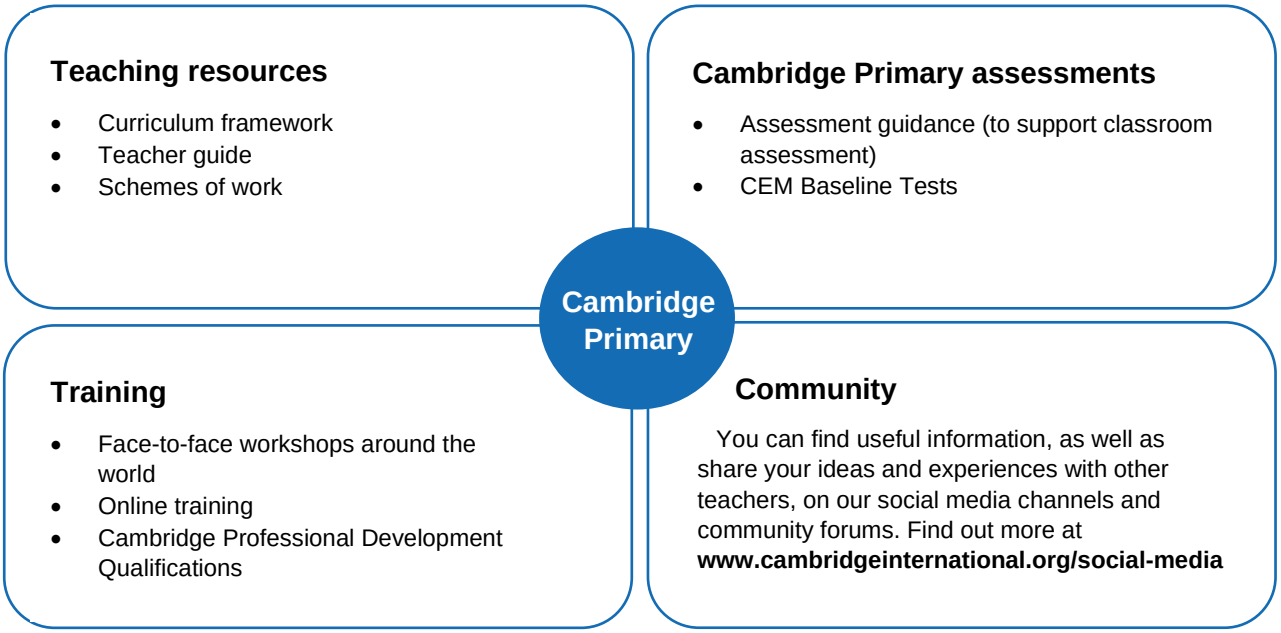
The Cambridge Primary Computing curriculum emphasises the increasing scale by which computers are used to control other devices. This provides a valuable opportunity for learners to explore the role of technology in industrial processes and in service industries. Overall, this

curriculum helps learners to understand how computers work and how they impact on local and global economies.

Supporting teachers

We provide a wide range of practical resources, detailed guidance, innovative training and professional development so that you can give your learners the best possible experience of Cambridge Primary Computing.

You will find most of these resources on the Cambridge Primary support site (**primary.cambridgeinternational.org**). Ask the Cambridge coordinator or exams officer in your school if you do not already have a log-in for this support site.



Progression through the Cambridge Pathway

Our primary programme is part of the Cambridge Pathway. This pathway leads seamlessly from primary to secondary and pre-university years. Each step of the pathway builds on the learners' development from the previous one or from other educational systems. This curriculum framework is typically for learners aged 5 to 11, but it may be appropriate to use it for slightly different ages to suit your context.

You can download more information on progression from the Cambridge Primary support site.

Teaching time

For guidance, this curriculum framework is based on learners having the following amount of teaching time for Computing per stage:

Stage	Hours per week	Hours per stage (based on 30 weeks of teaching)
1	0.75	22.5
2	1	30
3	1	30
4	1.5	45
5	1.5	45
6	1.5	45

Your actual number of teaching hours may vary according to your context.

2 Curriculum overview

Aims

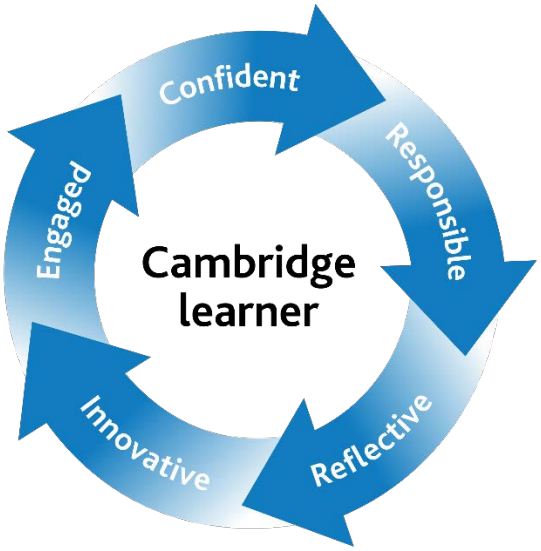
Following the Cambridge Primary programme helps learners to lay the foundations for lifelong learning, including:

- curiosity about the world around them and enthusiasm for learning
- knowledge, understanding and skills that can be applied in and across subjects
- effective and confident communication skills, including in English
- understanding of their personal and local context, as well as having global awareness.

In Cambridge Primary Computing, learners:

- become confident computational thinkers, who can abstract key information from a set of instructions, break down problems into smaller parts and recognise patterns within sequences of instruction. They can represent sequences of instructions both verbally and visually, with increasing precision.
- think logically, and identify and solve errors in increasingly complex computing scenarios.
- see themselves as computer scientists, who identify opportunities for skills such as programming and logical thinking in a range of local and global industries.
- understand the role that data plays in the lives of individuals, businesses and in the wider world. They also understand how to use computers to gather, store, sort and present data for a range of purposes.
- develop the vocabulary that is regularly associated with computers and with computational thinking.
- evaluate sequences of instructions and understand the value of working collaboratively so that a range of skills can be applied to the development of computer programs.
- understand how computers and other machines are interconnected and how they play a vital role in a range of industries.

The Cambridge approach encourages learners to be:



Cambridge Primary Computing supports learners to become:

Responsible – They fully engage with the devices they use, and the programs they create or adapt. They recognise that logical thinking is an important skill for computer scientists and for their own wider learning.

Innovative – They use their computing skills to think innovatively and creatively within Computing and other subjects. They understand that the hardware and software they will use as adults will be far more powerful than what they use today.

Confident – They develop the confidence to try new things and to learn from their mistakes when programming or creating sequences of instruction. They have the confidence to persevere in identifying the errors and in finding ways to resolve them.

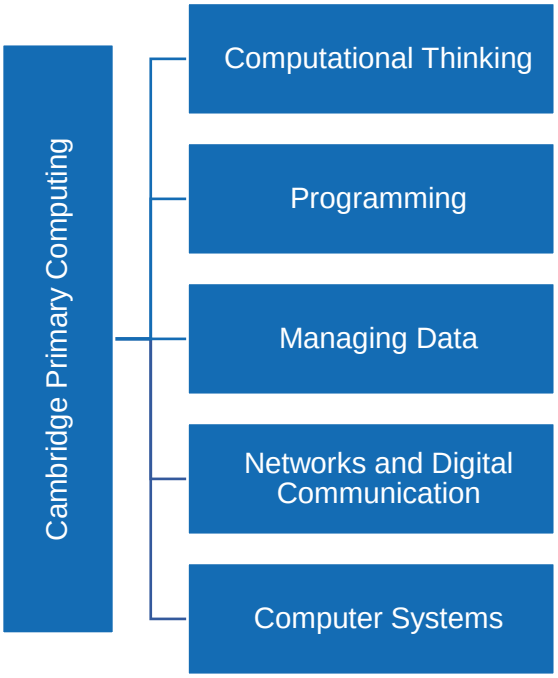
Engaged – They understand the strong relationships that exist between computers and data. They also understand the valuable role that computers play in collecting, storing, sorting and presenting data through a range of platforms including databases, spreadsheets and the World Wide Web.

Reflective – They reflect on the role that computers play in society and also in manufacturing and service industries. They recognise how computers and computational thinking will support them as they continue their education.

Overview of the strands

This curriculum framework provides a comprehensive set of learning objectives for Cambridge Primary Computing. These give a structure for teaching and learning and a reference against which learners' attainment and skills development can be checked.

We have divided the learning objectives into five main areas called 'strands' which run through every primary stage. The strands are listed in the following diagram:



Below is a brief description of each strand:

Computational Thinking

Learners understand the importance of clarity in the sets of instruction that they create, starting with simple sequences that describe tasks that they will already be familiar with. They will become increasingly aware that there is often more than one solution to a problem or task.

As learners move through the primary stages, they will begin to break tasks and problems into smaller, more manageable, parts, and identify repeated instructions to develop effective solutions.

Learners also investigate different ways of presenting sets of instruction and are able to explain that some methods are more effective than others. They also understand the value of working through algorithms systematically, to make predictions about outputs and to identify errors.

Programming

Learners understand how the sets of instructions to solve problems can be given to a computer. Their instructions may be simple and follow a linear sequence, but learners will begin to understand that the computer will only respond as expected if the instructions are precise and complete. Learners also begin to understand the benefits of testing their programs and of learning

from their mistakes. They will understand that debugging errors is an important part of a computer scientist's daily work.

Learners explore the role of computer scientists in everyday life, including how the instructions that are given to computers can be used to control other machines. They will investigate how programmers often work with a range of experts to refine their programs and to create a final product.

Learners' programs will use a range of constructs, which will provide a valuable starting point for the programming they will do in their later education and, possibly, in their working lives.

Managing Data

Learners consider the role of both data and information in their lives, as well as the differences between the two. They also investigate how computers are used to gather, store, search and analyse data, so that it can be presented as information.

Learners also explore how computers convert, process and respond to the data they receive through a range of inputs and sensors. They also work with a variety of software to gather and store data from different scenarios.

Networks and Digital Communication

This strand complements Cambridge Primary Digital Literacy and enables learners to understand the technical aspects behind the connectivity of computers and related hardware, both locally and globally. Learners consider:

- what connected computers enable us to do
- the technology that is required to enable this to happen and
- the safeguards that need to be in place to protect the data that is being transferred.

Learners also understand the role that global and local networks play in their lives and they draw a clear distinction between the internet and the World Wide Web.

Computer Systems

Learners understand how computers work, including how devices process inputs to create outputs. They also begin to understand how computers can be used to control other machines.

Learners explore how evolving technologies enable both individuals and entire industries to achieve new things that extend far beyond the tasks that personal digital devices are typically used for. They reflect on fictional portrayals of robots, along with the role robotics and artificial intelligence have in manufacturing and in service industries, such as health care.

Overview of teaching approaches

The Cambridge Primary Computing curriculum offers a broad range of activities that promote experience, reflection and improvement. Wherever possible, computational thinking should be taught in an integrated way, rather than focusing each lesson on a single element, such as sequencing or debugging. Lessons should include regular opportunities for learners to consider the order of the steps that they are following, or creating, for each task and whether these are precise.

Computational thinking, programming and data-related content should provide learners with an authentic and meaningful learning experience. Learners should enjoy the programs that they create, and they can do this through activities, such as:

- creating their own characters who respond to the instruction that the learners program, or by
- setting programming challenges for each other.

Data activities will be most effective if they are taught in contexts that the learners will enjoy, such as gathering data taken from polls of their peers.

Learners should be given regular opportunities to see computers, and their related hardware, in action. This will help them to understand contexts that are beyond their personal use of digital devices. For example, site visits or videos of automated manufacturing processes will assist learners to understand the connectivity of devices and how they combine to produce outputs such as manufactured products, sounds, decisions or information.

You can find more information and ideas for teaching and learning activities in the *Cambridge Primary Computing Teacher Guide* and schemes of work available on the Primary support site (**primary.cambridgeinternational.org**).

The teacher guide will support you to plan and deliver lessons using effective teaching and learning approaches.

The scheme of work for each stage of Cambridge Primary Computing contains:

- suggested units showing how the learning objectives in the curriculum framework can be grouped and ordered
- at least one suggested teaching activity for each learning objective
- a list of subject-specific language that will be useful for your learners
- additional support for aspects which may be unfamiliar to you and your learners, such as screenshots which explain the use of particular programming languages
- optional end of stage projects that enable learners to apply and consolidate their learning
- sample lesson plans.

You do not need to use the ideas in the schemes of work to teach Cambridge Primary Computing. They are designed to indicate the types of activities you might use, and the intended depth and breadth of each learning objective. These activities are not designed to fill all the teaching time for each primary stage. You should use other activities with a similar level of difficulty, for example, those from endorsed resources.

We work with a range of publishers to provide high-quality endorsed resources to support our curriculum frameworks. In order to provide choice for Cambridge International Schools, we

encourage publishers to develop resources with varying approaches. There is no requirement for endorsed textbooks to follow the teaching order suggested in the Cambridge Primary schemes of work. If a resource is endorsed, you can be confident that all the learning objectives are covered.

You should also consider the benefit of allowing learners repeated opportunities to practise certain skills, such as those related to programming. As an example, Stage 3 contains one learning objective about creating a program which has two algorithms running at once. As this is such an important programming concept, learners will benefit from repeated opportunities to practise and refine this in a range of different contexts, such as:

- a game or animation that includes two active characters, or
- a data program that produces different outputs depending on the input.

Teaching tools

The curiculum is designed to be used with a range of teaching tools that are both easy and affordable to access. Many activities do not require the use of any technology so learners can benefit from using traditional tools, such as pen and paper, when following, designing and presenting algorithms. In computing, activities that do not require the use of technology are known as ‘unplugged’.

Pre-prepared sets of cards are useful when teaching Cambridge Primary Computing. Learners can use these cards to place specific instructions into the correct sequence. Also, as vocabulary is an important part of this curriculum, key vocabulary cards should be displayed which contain computing words. These cards should be used to give learners regular opportunities to explain their understanding of the words.

When learners are creating programs on screen we recommend that they use Scratch Jr, **www.scratchjr.org**, in Stages 1 and 2 and Scratch, **scratch.mit.edu**, in Stages 3 to 6. Both Scratch platforms provide a range of information and ideas to support teachers and are available in many spoken and written languages. In Scratch, languages can be selected by clicking on the globe icon,  , in the top left corner of the ‘create’ screen.

We advise you to download the offline version of Scratch from **scratch.mit.edu/download** as this will enable the programming environment to be used when an internet connection is unavailable. Scratch Jr can be downloaded as an app, for free, from a range of app stores.

When using Scratch as a resource for teaching and learning it is a good idea to set up a ‘Scratch Classroom’ to manage learners’ accounts. This will enable you to view and comment on learners’ progress as they are working. The ‘Scratch classroom’ also provides options for sharing code with, and between, learners. This will enable learners to work collaboratively on certain coding tasks and provide an option for them to be given some partially completed code that they can edit. Detailed instructions for setting up Scratch Classroom can be found at **scratch.mit.edu/educators/faq**.

Learners are also required to use programmable devices from Stage 2 onwards. In Stage 2, this is likely to be a programmable toy, such as a beebot, that responds to directional instructions. If these toys are not available, an onscreen emulator can be used such as **beebot.terrapinlogo.com**. From Stage 3, we recommend that learners use a device such as the micro:bit, **microbit.org**. The micro:bit contains a number of input and output features including lights, buttons and sensors and

can be programmed using Scratch or Microsoft MakeCode, which is a block-based language that is similar to Scratch.

We recommend that enough computers and other programmable devices are available for learners to work in pairs or small groups.

Learners are likely to use a range of software packages in other subjects, for example for text processing or for creating presentations. The only specific software package that is required for Cambridge Primary Computing is one that allows learners to input, store and interact with data. We recommend that Google Forms is used for teaching the data content as it freely available and provides the functionality that learners will need when they are working with data.

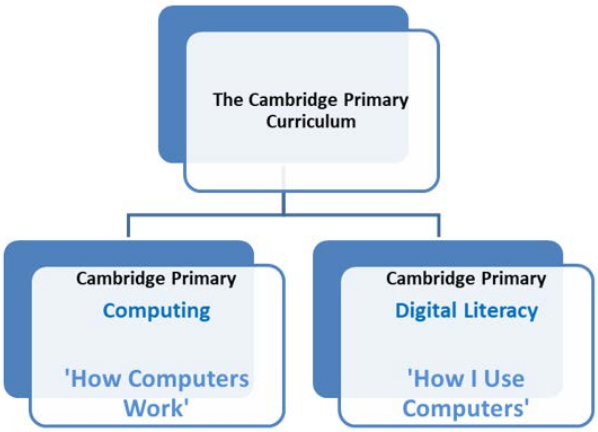
3 Computing and Digital Literacy

Content explanation

Cambridge Primary Computing supports learners to see inside a computer, to understand how computers work and how devices are programmed to communicate with each other and with other technological devices. This is different to the knowledge and skills related to the creation of digital outputs, such as typed documents, videos and multimedia, that they will learn through the Cambridge Primary Digital Literacy curriculum.

The Digital Literacy curriculum documents are available on the Cambridge Primary support site and these also support learners to develop responsibility for their personal safety and for their behaviour when online. Learners of Digital Literacy will also understand the impact that evolving technology is having on both individuals and on global communities.

The relationship between these two Cambridge Primary curricula can be explained as follows:



Primary learners do not need to follow both curricula but there are many benefits if they do.

The activities in Stage 1 of Cambridge Primary Digital Literacy assist young learners to log in and to interact with onscreen items, and to use a mouse or touchpad through developing their motor skills. Learners can then continue to refine these motor skills in their Computing lessons, for example by picking, dragging and dropping blocks of code into sequences of instruction.

Much of the content of our scheme of work for Stage 1 of Cambridge Primary Computing is ‘unplugged’ with a focus on the development of computational thinking skills, such as creating sequences of steps to perform everyday tasks. These activities allow time for learners to be developing the Digital Literacy skills that they will need for the later Computing content.

The Cambridge Primary Digital Literacy curriculum should be seen as an enabling curriculum for all areas of a learner’s education, and for their all round development, and not just be related to Computing or Computer Science.

4 Learning objectives by stage

Overview of learning objectives

There are learning objectives for each of Stages 1 to 6.

To enable effective progression in your teaching, you need to be familiar with the progression of skills across stages. This will help you to build on prior learning in every stage. The progression of learning objectives across Stages 1 to 6 is available on the Primary support site (**primary.cambridgeinternational.org**).

Learning objective codes

Each learning objective has a unique code, e.g. **1CT.05**. These codes appear in the schemes of work, teacher guide and other Cambridge Primary resources. Each learning objective code includes:

- the stage number, e.g. **1**
- a reporting code that reflects the strand titles, e.g. **CT** is **C**omputational **T**hinking
- a number reflecting the order of the learning objectives in the strand for the stage, e.g. **05** is the fifth learning objective.

Stage 1

Computational Thinking

- **1CT.01** Follow the steps in algorithms for everyday tasks.
- **1CT.02** Identify single errors in algorithms that represent everyday events or tasks.
- **1CT.03** Know how to give simple instructions, using directional language (forward, backwards, left, right), to navigate a path.
- **1CT.04** Suggest sets of ordered instructions to complete simple tasks, such as drawing a picture of a particular object or building a brick tower.
- **1CT.05** Know that an algorithm is a set of instructions to complete a task or to solve a problem.
- **1CT.06** Know that the order of instructions is important when creating algorithms.
- **1CT.07** Suggest ways that algorithms can be changed to affect the outcome.

Programming

- **1P.01** Know that algorithms can be recreated as code on computers.
- **1P.02** Know how to recreate algorithms as programs to perform simple tasks.
- **1P.03** Predict what is likely to happen when programs are run.
- **1P.04** Know that programs can contain errors.
- **1P.05** Know how to run programs to test whether they produce the desired result.
- **1P.06** Identify why a program does not produce the desired result.
- **1P.07** Know that 'debugging' is the correction of errors in programs.

Managing Data

- **1MD.01** Know that computing devices can be used in different ways to answer many different types of question.
- **1MD.02** Know that computing devices can help to sort and organise data.
- **1MD.03** Know how to use computing devices to manually record data, including using a form.
- **1MD.04** Identify questions that can be answered using data tables, limited to categorical data.

Networks and Digital Communication

- **1DC.01** Know that some devices can connect to each other to make a network.
- **1DC.02** Know that the internet consists of many computers that are connected together around the world.
- **1DC.03** Identify that some devices are connected by wires and that other devices are not.
- **1DC.04** Know that there are times when the internet is not available.

Computer Systems

- **1CS.01** Know that there are a range of computer systems with different functions, including communication, entertainment, creativity, research and for controlling other technology.
- **1CS.02** Know that computers can run many different programs, including games, apps and document creation tools.
- **1CS.03** Know that information and data can be input to computers in many different ways.
- **1CS.04** Know that computers can output information in many different ways.
- **1CS.05** Know that there are many everyday devices that use computers to control what they do.
- **1CS.06** Identify what robots are and where they may be found in the real world.

Stage 2

Computational Thinking

- **2CT.01** Follow and understand linear algorithms.
- **2CT.02** Identify and correct a single error in algorithms that represent everyday events or tasks.
- **2CT.03** Know that an algorithm is a precise set of instructions.
- **2CT.04** Identify the steps needed to undertake tasks, in order to develop simple algorithms.
- **2CT.05** Predict the outputs of algorithms.
- **2CT.06** Know how to develop precise sets of instructions to complete simple tasks, such as drawing a picture of a particular object or building a brick tower.

Programming

- **2P.01** Understand that programs instruct computers how to run algorithms.
- **2P.02** Know how to recreate algorithms as programs.
- **2P.03** Know how to develop programs to produce desired outputs, including the use of the repeat command.
- **2P.04** Know how to plan the instructions for objects within programs.
- **2P.05** Understand the benefits of working with others when debugging programs.
- **2P.06** Identify the benefits of regularly testing programs throughout their development.
- **2P.07** Know how to debug programs so that they will run and will produce the desired output.
- **2P.08** Know how to enter directional instructions in to a physical computing device to enable it to reach a specific destination.

Managing Data

- **2MD.01** Know the advantages of storing data and information on computers.
- **2MD.02** Know how to use computing devices to present categorical data.
- **2MD.03** Investigate different ways of using computing devices to collect categorical data for a particular purpose.
- **2MD.04** Identify types of statistical data that can be manually recorded using computing devices.
- **2MD.05** Discuss the different types of data that a question may generate, limited to statistical and non-statistical.
- **2MD.06** Understand how data may help to solve problems.

Networks and Digital Communication

- **2DC.01** Identify a range of devices that can connect to a network, including to the internet.
- **2DC.02** Know that two devices working together can achieve things that neither device can achieve on its own.
- **2DC.03** Know that there are wired and wireless networks.
- **2DC.04** Know how to recognise when a network is and is not available.
- **2DC.05** Know that network connected devices share information with each other and that there are risks associated with this.

Computer Systems

- **2CS.01** Use the correct terminology to explain the functions of basic hardware and software.
- **2CS.02** Identify some features that make digital devices easy to use, including their physical parts and their functions.
- **2CS.03** Know the difference between input and output devices.
- **2CS.04** Identify tasks that computers can complete more effectively than humans.
- **2CS.05** Understand that people use different types of computer device depending on a range of factors, including their location or their purpose.
- **2CS.06** Compare the representation of robots in fiction with real robots that have a real world purpose.

Stage 3

Computational Thinking

- **3CT.01** Follow, understand, edit and correct linear algorithms.
- **3CT.02** Understand that efficient algorithms are concise.
- **3CT.03** Identify steps that are repeated within everyday tasks.
- **3CT.04** Know that logical thinking is used in the creation of algorithms.
- **3CT.05** Predict the outcome of a change to an algorithm that is presented as a sequence of steps.
- **3CT.06** Know that many tasks can be divided into smaller sections to make them easier to follow and to edit.
- **3CT.07** Identify the inputs to algorithms.
- **3CT.08** Know how to develop linear algorithms to produce an output based on an input.

Programming

- **3P.01** Understand the benefit of editing programs to make them clear and concise, including removing unused commands or combining duplicated commands.
- **3P.02** Know how to develop programs that include code to reset objects to their original state (initialisation).
- **3P.03** Know how to create programs with more than one algorithm running at the same time.
- **3P.04** Know how to develop programs that contain more than one object, including a static object.
- **3P.05** Know how to make a change within a block of code to achieve desired outcomes in programs, such as changing the number of steps a sprite moves.
- **3P.06** Know how to create programs to produce an output from an input device.
- **3P.07** Outline the benefits of working with others when creating programs.
- **3P.08** Understand that programmers use their mistakes to inform the programs that they create.
- **3P.09** Know how to test and debug programs so that they run and produce the desired output.
- **3P.10** Know how to develop programs for a physical computing device to produce outputs.

Managing Data

- **3MD.01** Identify problems that can be solved through the collection and interpretation of data.
- **3MD.02** Identify and investigate different ways of representing discrete and categorical data, using a digital tool.
- **3MD.03** Know how to record discrete and categorical data, using computing devices.
- **3MD.04** Know that spreadsheets are comprised of rows and columns of cells and that data can be entered into the cells.
- **3MD.05** Know how to format cells according to their purpose, such as date, currency and text.
- **3MD.06** Demonstrate how to select data based upon their characteristics to solve problems.

Networks and Digital Communication

- **3DC.01** Identify networked hardware in a familiar environment, including the school and home.
- **3DC.02** Identify services that are available on familiar networks, including digital files, printed documents and the World Wide Web.
- **3DC.03** Understand the advantages and disadvantages of a network.
- **3DC.04** Know that ciphers are a way of making sure that information stays secret.
- **3DC.05** Know how to write and decode messages using very simple code, including converting letters to numbers (1 = a, 2 = b, etc.).

Computer Systems

- **3CS.01** Know that the hardware and software components of computing devices combine to form a working system.
- **3CS.02** Know the differences between hardware and software, and compare the different roles that they perform in computer systems.
- **3CS.03** Identify a range of manual and automatic input devices.
- **3CS.04** Know that different types of file can be stored on a computer's hard drive, including text, audio, image, video and games.
- **3CS.05** Know that computers can be programmed to control machines and other physical objects.
- **3CS.06** Identify common 'Internet of Things' devices in a familiar environment.
- **3CS.07** Explain the role of robots in manufacturing.

Stage 4

Computational Thinking

- **4CT.01** Follow, understand, edit and correct algorithms that use repetition, including indefinite (forever) loops.
- **4CT.02** Follow, understand, edit and correct algorithms that use iteration, including count-controlled loops.
- **4CT.03** Understand that the use of repetition can make algorithms more concise.
- **4CT.04** Compare and contrast algorithms designed for the same task to determine which produces the outcome that is best suited to the purpose.
- **4CT.05** Predict the outcome of algorithms that contain repetition.
- **4CT.06** Understand that decomposition is a process that is used to break tasks into different parts (sub-routines).
- **4CT.07** Know how to use decomposition to break tasks into different parts, represented as algorithms.
- **4CT.08** Follow and understand algorithms that use a sub-routine.
- **4CT.09** Know how to develop algorithms to produce different outputs based on different inputs.
- **4CT.10** Know how to develop algorithms that include repetition.

Programming

- **4P.01** Know how to add comments to blocks of code and explain the benefits of these comments.
- **4P.02** Know how to develop programs with repetition.
- **4P.03** Know how to develop programs with iteration.
- **4P.04** Know how to develop programs that produce a desired output, which includes the use of the repeat command.
- **4P.05** Know how to develop programs that produce different outputs from different inputs.
- **4P.06** Know how to plan the instructions for objects within programs, including identifying inputs and outputs.
- **4P.07** Know how to test different parts of a program systematically, to identify and debug errors.
- **4P.08** Know how to develop programs for a physical computing device to produce outputs from input devices.
- **4P.09** Know how to develop programs for a physical computing device using count-controlled and indefinite (forever) loops.

Managing Data

- **4MD.01** Understand the differences between physical (paper-based) and digital databases.
- **4MD.02** Understand the advantages and disadvantages of using forms when collecting data.
- **4MD.03** Identify the differences between data and information.
- **4MD.04** Know how to sort data into a required order, including descending or ascending numerical values and alphabetically.
- **4MD.05** Identify appropriate data types for a field within a data table.
- **4MD.06** Know how to use a database to answer a single question.
- **4MD.07** Identify data, records and fields within a data table.

Networks and Digital Communication

- **4DC.01** Explain the role of servers and clients in a network.
- **4DC.02** Describe the differences between the World Wide Web and the internet.
- **4DC.03** Describe the differences between wi-fi and ethernet, including speed, security and the use of wires.
- **4DC.04** Identify issues that may occur as a result of a failure in a network.
- **4DC.05** Identify where and why encryption is used in digital systems.
- **4DC.06** Know how to write and decode messages using the Caesar Cipher and the Pigpen Cipher.

Computer Systems

- **4CS.01** Identify examples where a control system is used.
- **4CS.02** Know the functions of application and systems software.
- **4CS.03** Identify a range of data recorded by input devices in computer systems, including data that is collected through sensors and data loggers.
- **4CS.04** Identify a range of information communicated by output devices in computer systems.
- **4CS.05** Identify that different types of file have different sizes, including text, audio, image, video and games.
- **4CS.06** Describe the role of computer scientists in a range of industries.
- **4CS.07** Identify the role of robots in service industries, including for delivery services, public transport and health care.

Stage 5

Computational Thinking

- **5CT.01** Follow, understand, edit and correct algorithms that contain selection.
- **5CT.02** Understand that different algorithms can be used to complete a task, with some algorithms being more efficient than others.
- **5CT.03** Predict the outcome of algorithms that contain a selection statement.
- **5CT.04** Understand and use variables in algorithms, including how to assign a variable to a specific value.
- **5CT.05** Know how to develop algorithms where two objects interrelate.
- **5CT.06** Understand and use selection in algorithms, limited to IF, THEN, ELSE.
- **5CT.07** Understand and use comparison (equal to) operators in algorithms.
- **5CT.08** Understand and use arithmetic (+, −) operators in algorithms.

Programming

- **5P.01** Understand the importance of creating a clear name for each variable.
- **5P.02** Know how to develop programs where two or more objects can interact.
- **5P.03** Know how to develop programs with a variable assigned to a specific value.
- **5P.04** Know how to develop programs with simple conditional, or selection, statements, including IF, THEN, ELSE, to produce different outputs.
- **5P.05** Know how to develop programs with 'equal to' comparison operators.
- **5P.06** Know how to develop programs with a variable modified using arithmetic operators.
- **5P.07** Know how to identify the purpose, events and expected outcomes to write an outline plan for a program.
- **5P.08** Know how to plan the instructions for objects within a program, including the identification of the outcomes of conditions and the data stored.
- **5P.09** Outline that the creation of a final program often requires input from people with a range of skills.
- **5P.10** Evaluate programs against given criteria.
- **5P.11** Know how to develop programs for a physical computing device with simple conditional statements, including IF, THEN, ELSE, to produce different outputs.

Managing Data

- **5MD.01** Know that a range of computing tools may be used during a statistical investigation, including data loggers, spreadsheets and databases, and document production tools.
- **5MD.02** Identify different ways of representing categorical, discrete, and continuous data, using computing devices.
- **5MD.03** Know how to modify representations of data to suit different criteria.
- **5MD.04** Know how to collect data for sets of related questions, limited to categorical and discrete data.
- **5MD.05** Know that cells can be restricted to accept only certain data types, limited to text, date and number.
- **5MD.06** Know how to use the arithmetic operators, including +, −, *, /, and simple functions, such as SUM and AVERAGE, in a spreadsheet.
- **5MD.07** Identify the impact of changes to individual data items.
- **5MD.08** Know how to identify data based on a single criterion, including data that matches a key word.

Networks and Digital Communication

- **5DC.01** Explain the role of switches, routers and wi-fi access points in a network.
- **5DC.02** Know that all devices on a network have an IP address which is used by data to identify its destination.
- **5DC.03** Explain how websites are stored on servers and accessed over the internet.
- **5DC.04** Explain the role of a cellular network when connecting mobile devices to the internet.
- **5DC.05** Know that data is divided into smaller pieces, transmitted as packets through multiple devices over networks and reassembled at the destination.
- **5DC.06** Know that packets can follow different routes across the internet before being reassembled.
- **5DC.07** Identify issues that may occur as a result of a failure of the internet.

Computer Systems

- **5CS.01** Describe the scale of the computing devices and mechanisms that are connected to the internet, either for data input or as actuators to make something happen, both in the home and globally.
- **5CS.02** Know that there are a range of storage devices that can be used within computer systems.
- **5CS.03** Know that computers represent data in binary (0,1).
- **5CS.04** Identify bits, bytes, kilobytes and megabytes, making links to memory size and storage.
- **5CS.05** Describe the input-process-output model and illustrate with examples relating to different devices, including control systems, printing and audio production.
- **5CS.06** Know that Artificial Intelligence (AI) is a simulation of human intelligence within computer systems.
- **5CS.07** Know that AI is used within common productivity software, limited to predictive text or speech to text.

Stage 6

Computational Thinking

- **6CT.01** Follow and understand algorithms that are presented as flowcharts.
- **6CT.02** Understand the symbols used in flowcharts, limited to start, stop, process, procedure (sub-routine), decision and the connector.
- **6CT.03** Predict the outcomes of flowcharts.
- **6CT.04** Know that variables can be used in different algorithms.
- **6CT.05** Know that the same sub-routine can be used multiple times in an algorithm.
- **6CT.06** Know that a sub-routine can be used in different algorithms.
- **6CT.07** Know how to develop algorithms that include two or more variables.
- **6CT.08** Understand and use arithmetic (+, −, *, /) operators in algorithms.

Programming

- **6P.01** Explain the use of constructs in programming, including sequence, selection and iteration.
- **6P.02** Know how to develop block-based programs with a procedure (sub-routine) to define commonly used sections of code.
- **6P.03** Know how to develop block-based programs where multiple algorithms interrelate.
- **6P.04** Know how to develop block-based programs using data types, including Integer, Character and String.
- **6P.05** Know how to develop block-based programs with combined constructs across multiple objects and that meet set criteria, including:
  - variables
  - conditionals (selection)
  - arithmetic and comparison operators
  - loops
  - procedures
  - interaction.
- **6P.06** Describe the role of prototypes when designing programs.
- **6P.07** Know how to develop prototypes of interfaces for programs, including suitable prompts for its users.
- **6P.08** Know how to follow project plans to develop programs.
- **6P.09** Define and use criteria to evaluate programs.
- **6P.10** Know how to test programs using a range of data.
- **6P.11** Know how to develop programs for a physical computing device to generate outputs based on a range of inputs, including the use of a variable.

Managing Data

- **6MD.01** Identify the role of different computing tools when planning statistical investigations.
- **6MD.02** Design appropriate forms to capture continuous data for given purposes.
- **6MD.03** Design spreadsheets that include a combination of features, including cell referencing, arithmetic operators and functions limited to SUM, and AVERAGE.
- **6MD.04** Select data that is relevant for particular purposes.
- **6MD.05** Design and create single table databases, including data attributes and data types, for given purposes.
- **6MD.06** Know how to use phrase searching to find information in databases.
- **6MD.07** Know that data is used to solve problems in a range of industries, including health, manufacture and retail.

Networks and Digital Communication

- **6DC.01** Know that a range of digital content is stored on servers, including streaming and messaging services.
- **6DC.02** Explain that digital devices can transfer data wirelessly using radio waves, including wi-fi and cellular networks.
- **6DC.03** Understand how bandwidth affects network performance.
- **6DC.04** Know that a network can become overloaded if there are too many devices connected to it.
- **6DC.05** Explain the need to keep data secure during transmission.
- **6DC.06** Describe the different types of user authentication, including password, fingerprint and facial recognition.

Computer Systems

- **6CS.01** Know how to select hardware and software components, while considering a range of factors such as functionality, cost, speed and aesthetics.
- **6CS.02** Know that there are many different programming environments, such as block-based and text-based, and that some are more appropriate to use in a given situation.
- **6CS.03** Explain that analogue data must be digitised (converted into a numerical form) for processing by a computer, as computers can only store, process and communicate digital information.
- **6CS.04** Identify nibbles, bits, bytes (kilobyte, megabyte, gigabyte, terabyte), making links to memory size and storage.
- **6CS.05** Explain the role of a processor within a computer.
- **6CS.06** Explain the role of primary and secondary storage within a computer.
- **6CS.07** Know that robots can work autonomously.
- **6CS.08** Identify benefits of using robotics in industry, such as car manufacturing or food production.

5 Glossary

This glossary is provided to support your understanding of the content of this curriculum framework. The definitions are intended to be sufficient to guide an informed reader.

Actuator – an output device responsible for creating real world movement, such as controlling a robotic arm in a manufacturing process.

Algorithm – a set of precise instructions for solving a problem.

Bandwidth – the data transfer capacity of a network. It is usually measured in bits per second (Bps).

Categorical data – data arising from measurements taken on a categorical (unordered discrete) variable. (Examples: learners' favourite colours: red, blue, green, yellow.)

Cellular network – a network that allows mobile phones to connect. The last link is wireless but allows devices to connect to a transmitter within the local area, or cell.

Character – the data type used to store a single letter within a variable in a program.

Cipher – an algorithm that is used for performing encryption or decryption.

Client – a piece of hardware or software that accesses a service within a network.

Code – the instructions that are input to a computer to produce a program.

Comparison operator – an operator that compares values to make a decision within a program.

Computer – a machine that can follow stored instructions. It is the computers within digital devices that enable those devices to operate.

Computer scientist – a professional who designs new software / hardware and develops new ways to use technology to solve problems.

Conditional statement – the instruction in a program that instructs it to perform different actions depending on whether the condition is 'true' or 'false'. Also known as a Selection statement.

Continuous data – data arising from measurements taken on a continuous variable. (Example: weight of food packets.)

Construct – the common building blocks that are used when designing programs.

Data attribute – a feature of a data item that enables it to be categorised.

Data logger – a device that uses sensors to record data over time.

Data type – the content of data within a variable, for example whether it is text or numeric.

Debug – the finding and fixing of errors within an algorithm or program.

Decomposition – the taking of a complex problem and breaking it down into smaller parts that are easier to solve.

Digital device – a physical piece of equipment that contains a computer or microcontroller, such as a tablet, smartphone or desktop/laptop computer.

Digital system – the combination of hardware, software and networks in performing a task or range of tasks.

Discrete data – data that can only take certain values (Example: number of peas in a pod.)

Ethernet – a popular form of network cable, used to carry broadband signals between digital devices.

Flowchart – a diagram that uses a standard set of symbols, including arrows, to represent each step of an algorithm. The symbols are used to represent different types of instruction.

Function – a predefined formula that performs calculations using values from other cells within a data table.

Hard drive – a storage device that is used to store long term data within a computer, such as its operating system, applications and personal documents and files.

Hardware – the physical components of digital devices and networks.

Initialisation – the use of an initial value for an object within a program so that the object always starts in the same way each time a program is run. (Example: a sprite that starts from the same screen position at the beginning of a game or animation.)

Input – data from outside of a computer that is entering into the digital device.

Integer – the data type for a whole number within a variable in a program.

Interface – the point of contact between a program and its user.

Internet – the global system of connected computers enabling people and devices to communicate and share information. Learners should be supported to understand the difference between the internet and the World Wide Web.

Internet of Things – physical objects that contain software, sensors and other technologies that enable them to connect to and communicate with other devices. (Example: a domestic oven that can be controlled from the owner's smartphone.)

IP address – a unique address that identifies a device on the internet or on a local network.

Iteration – a single pass through a set of instructions within an algorithm or program.

Learning objectives – statements from the curriculum framework of the expectations of knowledge, understanding and skills that learners will develop; they provide a structure for teaching and learning, and a reference against which to check learners' attainment and skills development.

Linear algorithm – an algorithm where each step is followed in a strict sequence, for example it does not contain any repetition or decisions.

Loop – a section of code that is repeated within a program.

Network – the computers and other connected hardware that make it possible to transfer data between digital devices.

Output – data or information that leaves a digital device.

Packet – a small part of a larger message that has been broken down for transfer over the internet. Packets are reassembled once they reach the destination device.

Physical (computing) device – a device that learners can program and interact with through a range of inputs, including sensors, and outputs, including LEDs.

Procedure – a small section of a program that performs a specific task in a specific way. Procedures can be used many times within a program and can be re-used in other programs.

Processor – the hardware that takes instructions from memory and executes them. The processor is sometimes called the central processing unit, or CPU.

Program – a set of instructions that are encoded so that they can be understood by a computer so that it can produce an output based upon an input or on stored data.

Prototype – an early example of a product, including a device or a piece of software. Prototypes allow for early feedback and help to define the requirements of the final product.

Repetition – a section of code that is run a number of times within a program.

Robotics – a combination of computer science and engineering that uses computers to control robots to perform specific functions, for example within dangerous environments or within manufacturing.

Scheme of work – support materials for each stage of Cambridge Primary Computing. Each scheme of work contains a suggested long-term plan, a medium-term plan with suggested teaching and learning activities and sample short-term (lesson) plans.

Selection – a programming construct where a decision is made within a program, such as where the program decides to move in a particular direction based upon the outcome of a prior event or upon an input from a user.

Selection statement – the instruction in a program that tells it to perform different actions depending on whether the condition is ‘true’ or ‘false’. Also known as a Conditional statement.

Server – a device that provides services, such as storage or web pages, within a network.

Software – the programs that either control or are run on a computer.

Sprite – an object, such as an onscreen character, that can be programmed independently of other objects.

Statistical data – numerical data that can be collected, represented and interpreted.

Strand – a collection of learning objectives in the curriculum framework that forms an area of learning.

String – a data type that is used to represent sequences of letters, words and text within a variable. A string can contain a set of characters, including letters (or whole words), spaces and numbers.

Sub-routine – a discrete section of code that performs a particular operation within a program. Sub-routines can be used many times within a program and can be re-used in other programs.

Switch – hardware that enables multiple devices to be connected to the same network and is able to select specific destination devices for the data that is being received.

Teacher guide – a document providing support in using the curriculum framework to plan and deliver lessons using effective teaching and learning approaches.

Unplugged – computational thinking activities that take place without digital devices, such as following an algorithm through physical movement.

User authentication – a process that enables a device to check the identity of someone that is requesting access to that device or to another network resource.

Variable – a memory location within a program that stores particular values, such as the name, age or score of a user. The values can change each time the program is run or while the program is running.

Vocabulary – words and phrases.

Wi-fi – network technology that allows devices to connect to the internet without wires.

Wired / Wireless – describes whether devices are connected to a network with or without cables.

World Wide Web – a service provided by computers that are connected to the internet, consisting of pages of information that are transmitted to users upon request.

We are committed to making our documents accessible in accordance with the WCAG 2.1 Standard. We are always looking to improve the accessibility of our documents. If you find any problems or you think we are not meeting accessibility requirements, contact us at **info@cambridgeinternational.org** with the subject heading: Digital accessibility. If you need this document in a different format, contact us and supply your name, email address and requirements and we will respond within 15 working days.

Cambridge Assessment International Education
The Triangle Building, Shaftesbury Road,
Cambridge, CB2 8EA, United Kingdom

t: +44 (0)1223 553 554

www.cambridgeinternational.org

5690247428